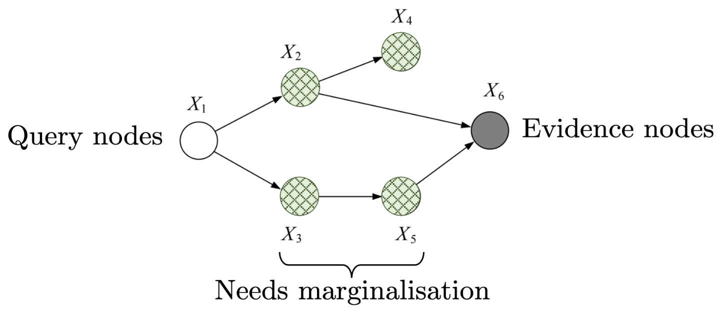
CS5340 Uncertainty Modelling in AI

AY2022/23 Semester 2 · Prepared by Tian Xiao *@snoidetx*

*CS5340 is about how to **represent** and **reason** with **uncertainty** in a computer.*

5 Variable Elimination & Belief Propagation



Variable Elimination: Use **factorization** and **distributive law** to reduce computational complexity:

$$
\begin{aligned}
& p(x_1, \overline{x}_6) \\
&= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(\overline{x}_6|x_2,x_5) \\
&= p(x_1)\sum_{x_2} p(x_2|x_1)\sum_{x_3} p(x_3|x_1)\sum_{x_4} p(x_4|x_2)\sum_{x_5} p(x_5|x_3)p(\overline{x}_6|x_2,x_5) \\
&= p(x_1)\sum_{x_2} p(x_2|x_1)\sum_{x_3} p(x_3|x_1)\sum_{x_4} p(x_4|x_2)m_5(x_2,x_3) \\
&= p(x_1)\sum_{x_2} p(x_2|x_1)\sum_{x_3} p(x_3|x_1)m_5(x_2,x_3)\sum_{x_4} p(x_4|x_2) \\
&= p(x_1)\sum_{x_2} p(x_2|x_1)\sum_{x_3} p(x_3|x_1)m_5(x_2,x_3)m_4(x_2) \\
&= p(x_1)\sum_{x_2} p(x_2|x_1)m_4(x_2)\sum_{x_3} p(x_3|x_1)m_5(x_2,x_3) \\
&= p(x_1)\sum_{x_2} p(x_2|x_1)m_4(x_2)m_3(x_1,x_2) \\
&= p(x_1)m_2(x_1).
\end{aligned}
$$

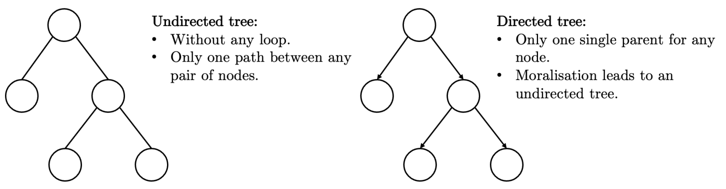
- In this way, $p(x_1|\overline{x}_6) = \frac{p(x_1)m_2(x_1)}{\sum_{x_1} p(x_1)m_2(x_1)}$.
- For UGMs, we do the same thing except changing local conditional probabilities into potentials of cliques.
  - $p(x_1|\overline{x}_6) = \frac{\frac{1}{Z}m_2(x_1)}{\frac{1}{Z}\sum_{x_1} m_2(x_1)}$. The normalization factor Z in conditional probabilities, but **not** in marginal probabilities.
- Computational complexities: Analyzed through reconstituted graphs.
  - For UGMs, for each node X_i, we connect all the remaining neighbours of X_i and remove X_i.
  - For DGMs, for each node X_i, we connect all the parents of X_i. In the end we drop the orientation of all edges (**moralisation**) and analyze it like UGMs.
  - Overall complexity: $O(nk^M)$, where M is the size of largest elimination clique.
  - Treewidth: One less than the smallest achievable cardinality of the largest elimination clique over all possible elimination orderings (NP-hard).
  - Heuristics:
    - ∗ Min-neighbors: Fewest number of dependent variables.
    - ∗ Min-weight: Minimize product of cardinalities of variables.
    - ∗ Min-fill: Minimize the size of the factor (elimination clique) that will be added.
- Limitations:
  - We have to re-run the variable elimination algorithm with every new query node.

Variable Elimination Algorithm

```
ELIMINATE($\mathcal{G}, E, F$)
    INITIALIZE($\mathcal{G}, F$)
    EVIDENCE($E$)
    UPDATE($\mathcal{G}$)
    NORMALIZE($F$)

INITIALIZE($\mathcal{G}, F$)
    choose an ordering $I$ such that $F$ appears last
    for each node $X_i$ in $\mathcal{V}$
        place $p(x_i \,|\, x_{\pi_i})$ on the active list
    end

EVIDENCE($E$)
    for each $i$ in $E$
        place $\delta(x_i, \bar{x}_i)$ on the active list
    end

UPDATE($\mathcal{G}$)
    for each $i$ in $I$
        find all potentials from the active list that reference $x_i$ and remove them from the active list
        let $\phi_i(x_{T_i})$ denote the product of these potentials
        let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
        place $m_i(x_{S_i})$ on the active list
    end

NORMALIZE($F$)
    $p(x_F \,|\, \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)$
```

When interested in MAP of query node, we use

$$m_i^{\max}(x_{S_i}) = \max_{x_i} \phi_i^{\max}(x_{T_i})$$



Undirected tree:
- Without any loop.
- Only one path between any pair of nodes.

Directed tree:
- Only one single parent for any node.
- Moralisation leads to an undirected tree.

Belief Propagation: To obtain all marginals in the tree, we reuse messages to perform efficient inference.

- Works similarly for undirected and directed trees.
- Messages:

$$m_{ji}(x_i) = \sum_{x_j}\left(\psi^E(x_j)\psi(x_i,x_j)\prod_{k\in\mathcal{N}(j)\setminus i} m_{kj}(x_j)\right)$$

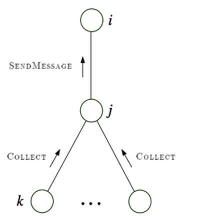
$$p(x_f|\overline{x}_E) \propto \psi^E(x_f)\prod_{e\in\mathcal{N}(f)} m_{ef}(x_f).$$

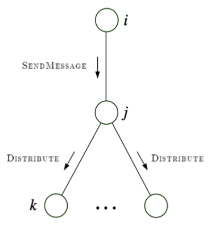
- Message-passing protocol: A node can send a message to a neighbouring node when and only when it has received messages from all of its other neighbours.
- MAP configurations: We record the maximising values in a table $\delta_{ji}(x_i)$ when a message $m_{ji}^{\max}(x_i)$ is sent from X_j to X_i (closer to root). We then use this table to define a consistent maximising configuration during an outward pass.
- Products of probabilities tend to underflow. We overcome this by using the monotone log scale:

$$\max_x p^E(x) = \max_x \log p^E(x).$$

Sum-Product Algorithm (Belief Propagation)

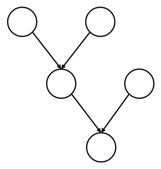
```
SUM-PRODUCT($\mathcal{T}$, $E$)
    EVIDENCE($E$)
    $f$ = CHOOSEROOT($\mathcal{V}$)
    for $e \in \mathcal{N}(f)$
        COLLECT($f, e$)
    for $e \in \mathcal{N}(f)$
        DISTRIBUTE($f, e$)
    for $i \in \mathcal{V}$
        COMPUTEMARGINAL($i$)

EVIDENCE($E$)
    for $i \in E$
        $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$
    for $i \notin E$
        $\psi^E(x_i) = \psi(x_i)$

COLLECT($i, j$)
    for $k \in \mathcal{N}(j) \backslash i$
        COLLECT($j, k$)
    SENDMESSAGE($j, i$)

DISTRIBUTE($i, j$)
    SENDMESSAGE($i, j$)
    for $k \in \mathcal{N}(j) \backslash i$
        DISTRIBUTE($j, k$)

SENDMESSAGE($j, i$)
    $m_{ji}(x_i) = \sum_{x_j}(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j)\backslash i} m_{kj}(x_j))$

COMPUTEMARGINAL($i$)
    $p(x_i) \propto \psi^E(x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i)$
```





Max-Product Algorithm

```
MAX-PRODUCT($\mathcal{T}$, $E$)
    EVIDENCE($E$)
    $f$ = CHOOSEROOT($\mathcal{V}$)
    for $e \in \mathcal{N}(f)$
        COLLECT($f, e$)
    $MAP = \max_{x_f}(\psi^E(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}^{\max}(x_f))$
    $x_f^* = \arg\max_{x_f}(\psi^E(x_f) \prod_{e \in \mathcal{N}(f)} m_{ef}^{\max}(x_f))$
    for $e \in \mathcal{N}(f)$
        DISTRIBUTE($f, e$)

COLLECT($i, j$)
    for $k \in \mathcal{N}(j) \backslash i$
        COLLECT($j, k$)
    SENDMESSAGE($j, i$)

DISTRIBUTE($i, j$)
    SETVALUE($i, j$)
    for $k \in \mathcal{N}(j) \backslash i$
        DISTRIBUTE($j, k$)

SENDMESSAGE($j, i$)
    $m_{ji}^{\max}(x_i) = \max_{x_j}(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j)\backslash i} m_{kj}^{\max}(x_j))$
    $\delta_{ji}(x_i) \in \arg\max_{x_j}(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j)\backslash i} m_{kj}^{\max}(x_j))$

SETVALUE($i, j$)
    $x_j^* = \delta_{ji}(x_i^*)$
```



Polytree:
- Nodes with more than 1 parent.
- Moralisation leads to loops.

6 Factor Graph & Junction Tree

Factor Graph: Introduce additional nodes for the factors.

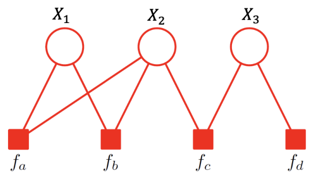
- Works for polytrees.
- A factor graph is a bipartite graph $\mathcal{G}(\mathcal{V}, \mathcal{F}, \mathcal{E})$, where
  - $\mathcal{V}$ is the set of random variables;
  - $\mathcal{F}$ is the set of factors. In DGM, all the local conditional distributions $p(x_i|x_{\pi_i})$ are represented as factors; in UGM, all the potential functions of cliques are represented as factors; normalising coefficients $\frac{1}{Z}$ is a factor defined over the empty set of variables;
  - $\mathcal{E}$ is the set of all undirected edges.
- Two types of messages:
  - Messages ν flow from variable to factor nodes:

$$\nu_{is}(x_i) = \prod_{t \in \mathcal{N}(i)\backslash s} \mu_{ti}(x_i).$$

    At leaf variable nodes, $\nu_{is} = 1$.
  - Messages μ flow from factor to variable nodes:

$$\mu_{si}(x_i) = \sum_{X_{\mathcal{N}(s)\backslash i}} \left(f_s\left(x_{\mathcal{N}(s)}\right) \prod_{j \in \mathcal{N}(s)\backslash i} \nu_{js}(x_j) \right)$$

    At leaf factor nodes, $\mu_{is} = f_s(x_i)$.
- Message-passing protocol: A node can send a message to a neighbouring node when and only when it has received messages from all of its other neighbours (for both variable and factor nodes).
- $m_{ji}(x_i)$ in UGM is equal to $\mu_{si}(x_i)$ in the factor graphs.



$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

Sum-Product Algorithm for Factor Graphs

```
SUM-PRODUCT($\mathcal{T}$, $E$)
    EVIDENCE($E$)
    $f$ = CHOOSEROOT($\mathcal{V}$)
    for $s \in \mathcal{N}(f)$
        $\mu$-COLLECT($f, s$)
    for $s \in \mathcal{N}(f)$
        $\nu$-DISTRIBUTE($f, s$)
    for $i \in \mathcal{V}$
        COMPUTEMARGINAL($i$)

$\mu$-COLLECT($i, s$)
    for $j \in \mathcal{N}(s) \backslash i$
        $\nu$-COLLECT($s, j$)
    $\mu$-SENDMESSAGE($s, i$)

$\nu$-COLLECT($s, i$)
    for $t \in \mathcal{N}(i) \backslash s$
        $\mu$-COLLECT($i, t$)
    $\nu$-SENDMESSAGE($i, s$)

$\mu$-DISTRIBUTE($s, i$)
    $\mu$-SENDMESSAGE($s, i$)
    for $t \in \mathcal{N}(i) \backslash s$
        $\nu$-DISTRIBUTE($i, t$)

$\nu$-DISTRIBUTE($i, s$)
    $\nu$-SENDMESSAGE($i, s$)
    for $j \in \mathcal{N}(s) \backslash i$
        $\mu$-DISTRIBUTE($s, j$)

$\mu$-SENDMESSAGE($s, i$)
    $\mu_{si}(x_i) = \sum_{x_{\mathcal{N}(s)\backslash i}} (f_s(x_{\mathcal{N}(s)}) \prod_{j \in \mathcal{N}(s)\backslash i} \nu_{js}(x_j))$

$\nu$-SENDMESSAGE($i, s$)
    $\nu_{is}(x_i) = \prod_{t \in \mathcal{N}(i)\backslash s} \mu_{ti}(x_i)$

COMPUTEMARGINAL($i$)
    $p(x_i) \propto \nu_{is}(x_i)\mu_{si}(x_i)$
```

Junction Tree: Probability distributions corresponding to loopy undirected graphs can be reparametrised as trees.

- Cluster graphs:
  - Nodes are clusters $C_i \subseteq \{X_1, \cdots, X_n\}$ where X_i are the random variables.
  - Edge between C_i and C_j associated with sepset $S_{ij} = C_i \cap C_j$.
- **Family Preservation**: Given a set of potentials $\Psi \in \{\psi_1, \cdots, \psi_k\}$ from an UGM, we assign each ψ_k to a cluster $C_{\alpha(k)}$ such that Scope$[\psi_k] \subseteq C_{\alpha(k)}$.
- Cluster potential: $\phi_i(C_i) = \prod_{k:\alpha(k)=i} \psi_k$.
- **Running Intersection Property**: For each pair of clusters C_i, C_j and variables $X \in C_i \cap C_j$, there exists an unique path between $C_i \cap C_j$ for which all clusters and sepsets contain X.

- Equivalently, for any X, the set of clusters and sepsets containing X from a tree.

- Cluster tree: A cluster graph without cycles is known as the a cluster tree.

- Clique trees (junction trees): A cluster tree that satisfies the running intersection property is called a clique tree or junction tree.

- Construction of junction trees:

  1. Triangulation via graph elimination.
  2. Obtain clusters and all possible sepsets: use elimination cliques as clusters.
  3. Assign cluster potentials.
  4. Get clique tree: find the maximum spanning tree with cardinality of sepsets as weight of edges. **A cluster tree is a clique tree only if it is a maximal spanning tree.**

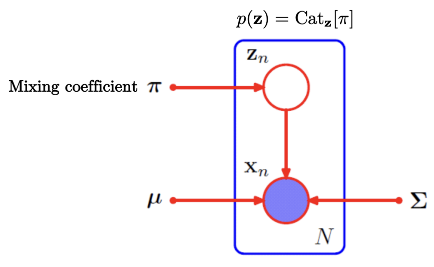
7 Mixture Models & Expectation Maximisation

Gaussian Mixture Models (GMMs): GMMs can approximate almost any continuous density with arbitrary accuracy. It is a linear combination of Gaussian distributions:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

- Each Gaussian density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is called a component of the mixture, and has its own mean $(\boldsymbol{\mu}_k)$ and covariance $(\boldsymbol{\Sigma}_k)$.

- The parameters $0 \leq \pi_k \leq 1$ is the mixing coefficients, and must sum to one: $\sum_{k=1}^{K} \pi_k = 1$.



$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$

- **Responsibility**: Measures the responsibility that component k takes for explaining the observation x:

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(\mathbf{x}|z_k = 1)p(z_k = 1)}{\sum_{j=1}^{K} p(\mathbf{x}|z_k = 1)p(z_k = 1)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}.$$

Expectation Maximisation for GMMs: MLE for GMMs does not have a closed form solution, hence we derive an iterative solution.

1. **Initialize** the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **Expectation Step**: Evaluate the responsibilities $\gamma(Z)$ using the current parameter values.
3. **Maximization Step**: Re-estimate the parameters using the current responsibilities.
4. Evaluate the log-likelihood and check for convergence.

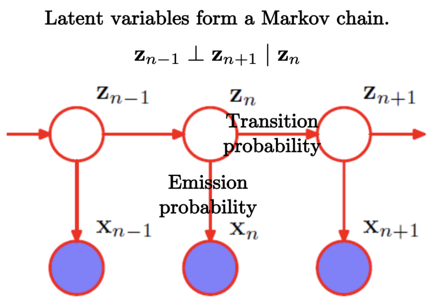
General EM Algorithm: To find maximum likelihood solutions for models having latent variables:

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \ln \int_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}).$$

1. Choose an **initial setting** for the parameters θ^{old}.
2. **Expectation Step**: Evaluate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})$.
3. **Maximisation Step**: Evaluate $\boldsymbol{\theta}^{\text{new}}$ given by $\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$, where $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$.
4. Check for convergence of either the log likelihood or the parameter values, if not converged: $\boldsymbol{\theta}^{\text{new}} \rightarrow \boldsymbol{\theta}^{\text{old}}$.

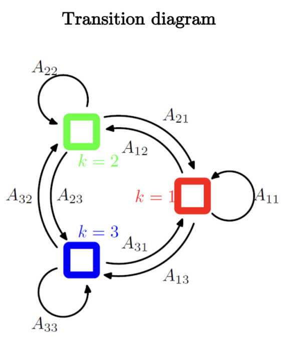
- **Theory behind EM**: $\ln p(\mathbf{X}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q\|p)$, where
  - $\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} \right\}$;
  - $\mathrm{KL}(q\|p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} \geq 0$.

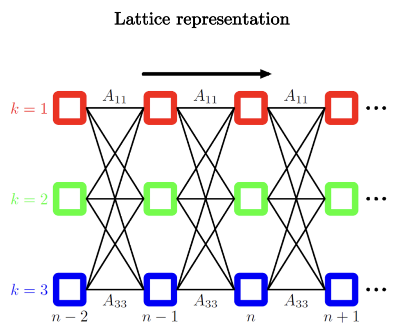
8 Hidden Markov Models



Hidden Markov Models: The Markov chain of latent variables gives rise to the graphical structure known as a state space model, where the joint distribution is given by

$$p(\mathbf{x}_1, \cdots, \mathbf{x}_n, \mathbf{z}_1, \cdots, \mathbf{z}_n) = p(\mathbf{z}_1) \left[\prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}) \right] \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n).$$





- Latent variables are discrete; observed variables can be either discrete or continuous.

- Transition probabilities: 1-of-K coding scheme for the discrete latent variables Z_n, which describes which mixture component is responsible for generating the observation X_n.
  - $p(z_n|z_{n-1})$ corresponds to a $K \times K$ matrix $\mathbf{A}$ with the following properties:
    1. $A_{jk} = p(z_{nk} = 1|z_{n-1,j} = 1)$.

2. $0 \leq A_{jk} \leq 1$, with $\sum_k A_{jk} = 1$.

3. $K(K-1)$ independent parameters.

▶ $p(\mathbf{z}_n|\mathbf{z}_{n-1}, \mathbf{A}) = \prod_{k=1}^K \prod_{j=1}^K A_{jk}^{z_{n-1,j} z_{nk}}$.

▶ Initial latent variable Z_1 does not have a parent node. It is represented as a categorical distribution $p(\mathbf{z}_1|\boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_{1k}}$, where $\sum_k \pi_k = 1$.

- Emission probabilities: $p(x_n|z_n, \phi)$, where ϕ is a set of parameters governing the distribution.

  ▶ $p(\mathbf{x}_n|\mathbf{z}_n, \phi) = \prod_{k=1}^K p(\mathbf{x}_n|\phi_k)^{z_{nk}}$.

- Homogenous model: Only 1 $\mathbf{A}$ and ϕ.

Expectation Maximisation for HMMs:

- Marginal posterior distribution: $\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.

- Joint posterior distribution: $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.

1. **E Step**: We use **forward-backward algorithm**:

Forward-Backward Algorithm

$$\gamma(\mathbf{z}_n) = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = \frac{\alpha(\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{n-1})\beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1})p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n)$$

Forward-Backward Algorithm (Rescaled)

$$\begin{aligned}\gamma(\mathbf{z}_n) &= \widehat{\alpha}(\mathbf{z}_n)\widehat{\beta}(\mathbf{z}_n)\\ \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= c_n\widehat{\alpha}(\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{-1})\widehat{\beta}(\mathbf{z}_n)\end{aligned}$$

$$c_n\widehat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}} \widehat{\alpha}(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

$$c_{n+1}\widehat{\beta}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \widehat{\beta}(\mathbf{z}_{n+1})p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n)$$

2. **M Step**: Find $\boldsymbol{\theta}$ that maximises

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})\\ &= \sum_{k=1}^K \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^N \sum_{j=1}^K \sum_{k=1}^K \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} +\\ &\quad \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \ln p(\mathbf{x}_n|\phi_k).\end{aligned}$$

- **Viterbi algorithm**: Max-sum algorithm in factor trees.

9 Monte-Carlo Inference

Monte-Carlo Sampling: Approximate a hard combinatorial problem by a much simpler problem using randomness.

- Monte-Carlo sampling is unbiased, consistent and converges at rate $\frac{1}{\sqrt{N}}$.

- Rejection sampling

- Importance sampling

Markov Chain Monte Carlo: Features adaptive proposals.

- Metropolis Hasting algorithm:

Algorithm : Metropolis-Hasting

```
1.  Initialize $x^{(0)}$
2.  For $i = 0$ to $N - 1$
3.        Sample $u \sim \mathcal{U}_{[0,1]}$            // draw acceptance threshold
4.        Sample $x' \sim q(x'|\, x^{(i)})$          // draw from proposal
5.        If $u < \mathcal{A}(x', x^{(i)}) = \min\left\{1, \frac{\tilde{p}(x')q(x^{(i)}|x')}{\tilde{p}(x^{(i)})q(x'|x^{(i)})}\right\}$   // acceptance probability
6.              $x^{(i+1)} = x'$           // new sample is accepted
7.        else
8.              $x^{(i+1)} = x^{(i)}$          // new sample is rejected
                                     // we create a duplicate of the previous sample
```

- **Ergodic Theorem** for Markov chains: If $X_0, \cdots, X_N$ is irreducible, homogenous, apeiodic discrete Markov chain with stationary distribution π, then

$$\frac{1}{N}\sum_{i=1}^N f(X_i) \to \mathbb{E}[f(X)] \text{ as } N \to \infty$$

almost for sure where $X \sim \pi$, and

$$p(x_N = x|x_0) \to \pi(x)\ \forall x, x_0 \in \mathcal{X} \text{ as } N \to \infty.$$

▶ Stationary distribution: $\pi T = \pi$.

▶ Limiting distribution: Markov chain always converges to π.

▶ Irreducibility: For any state of the Markov chain, there is a positive probability of visiting all other states.

▶ Aperiodicity: The Markov chain should not get trapped in circles.

▶ Ergodicity: A Markov chain is ergodic if it is irreducible and aperiodic.

▶ Detailed balance: A probability vector $\pi = p(x)$ satisfies detailed balance w.r.t. T if

$$\pi_a T_{ab} = \pi_b T_{ba}, \forall a, b \in \mathcal{X}.$$

Note that detailed balance implies stationary distribution.

Gibbs Sampling: A special case of the Metropolis Hasting algorithm where the acceptance probability is always one.

Algorithm : Gibbs Sampling

```
1.  Initialize $\{x_i : i = 1, \ldots, M\}$
2.  For $\tau = 1, \ldots, T$ :
3.      Sample $x_1^{\tau+1} \sim p(x_1|x_2^{(\tau)}, x_3^{(\tau)}, \ldots, x_M^{(\tau)})$.
4.      Sample $x_2^{\tau+1} \sim p(x_2|x_1^{(\tau+1)}, x_3^{(\tau)}, \ldots, x_M^{(\tau)})$.
5.      Sample $x_j^{\tau+1} \sim p(x_j|x_1^{(\tau+1)}, \ldots, x_{j-1}^{(\tau+1)}, x_{j+1}^{(\tau)}, \ldots, x_M^{(\tau)})$.
6.      Sample $x_M^{\tau+1} \sim p(x_M|x_1^{(\tau+1)}, x_2^{(\tau+1)}, \ldots, x_{M-1}^{(\tau+1)})$
```